VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Server pro podporu výuky ARUO - využití řešičů SAT

Supporting Server for Constraint Processing Teaching - Use of SAT Solvers

2015 Ivana Skýbová

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Ivana Skýbová**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Server pro podporu výuky ARUO - využití řešičů SAT
Supporting Server for Constraint Processing Teaching - Use of SAT solvers

Zásady pro vypracování:

Cílem bakalářské práce je vytvořit server, na kterém si uživatel bude moci přehledně zobrazit formuli výrokové logiky (tedy vstup pro problém SAT) odpovídající uživatelem zadaným vstupům zvolených úloh.

1. Seznamte se s řešiči SAT a standardními formáty vstupů, které akceptují.
2. Vytvořte výukový server, kde bude možné zadat vstupní data pro alespoň 3 různé problémy s omezeními a zobrazí se příslušná formule (vstup pro problém SAT) i s popisem, jak formule ze zadané instance problému vznikla.
3. Vytvořené formule umožněte uživateli stáhnout ve formátu DIMACS CNF.
4. Umožněte uživateli vložit řešení (tedy ohodnocení proměnných) získané spuštěním SAT solveru na stažený DIMACS CNF a převeďte toto řešení do pojmů původního problému.

Seznam doporučené odborné literatury:

[1] Rina Dechter: Constraint Processing, Morgan Kaufmann Publishers 2003, ISBN: 978-1-55860-890-0

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Martin Šurkovský**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

V Ostravě 4. května 2015 ..... Skýbová .....

Dovolím si na tomto místě poděkovat všem lidem, kteří mi svou nekonečnou trpělivostí a podporou pomohli tuto práci vytvořit. Jmenovitě si dovoluji poděkovat vedoucímu bakalářské práce Ing. Martinu Šurkovskému za cenné rady a připomínky.

Abstrakt

Práce pojednává o praktickém použití řešičů SAT, převodu problému na problém a implementaci serveru pro podporu výuky předmětu Automatizované řešení úloh s omezeními. Server je určen především pro studenty navazujícího magisterského studia oboru Informační a výpočetní technika, kteří si zvolili předmět ARUO a případně dalším zájemcům o problematiku, kterou se předmět zabývá. Pro implementaci byla zvolena technologie ASP.NET, aplikace je programovaná v jazyce C#, drobné skripty na straně klienta vykonává JavaScript. Aktuálně řešené problémy jsou: sudoku, úplné párování grafu a součet podmnožiny.

Klíčová slova: ARUO, SAT, řešič SAT, převod problému na problém, výukový server

Abstract

Thesis describes basic usage of SAT solvers, reduction between problems and implementation of a supporting server application for Constraint Processing teaching. Server is intended mainly to students of a Master Program with specialization in Computer Science and Technology, who choose to study Constraint Processing and eventually to others interested in this subject. It is implemented on the ASP.NET platform using programming language C#; minor client scripts are run by JavaScript. Currently solved problems are: sudoku, bipartite matching and subset sum.

Keywords: Constraint Processing, SAT, SAT solver, reduction between problems, educational server

Seznam použitých zkratek a symbolů

ARUO	–	Automatizované řešení úloh s omezeními
ASP.NET	–	Framework pro webovou aplikaci běžící na serveru
CSS	–	Kaskádové styly (Cascading Style Sheets)
CNF	–	Konjunktivní normální forma
CRLF	–	Konce řádků typické pro platformy Dos/Windows (Carriage Return, Line Feed)
DIMACS	–	The Center for Discrete Mathematics and Theoretical Computer Science
HTML	–	HyperText Markup Language
JavaScript	–	Skriptovací programovací jazyk běžící na straně klienta
JS	–	Viz JavaScript
KNF	–	Viz CNF
MVP	–	Návrhový vzor model-view-presenter
SAT	–	Splnitelnost (Satisfiability)
W3C	–	World Wide Web Consortium

Obsah

1 Úvod

Zadáním bakalářské práce byla implementace serveru pro podporu výuky předmětu ARUO. Tento text popisuje, jak se zadání podařilo splnit.

Server je určen především pro studenty navazujícího magisterského studia oboru Informační a výpočetní technika, kteří si zvolili předmět ARUO a případně dalším zájemcům o problematiku, kterou se předmět zabývá.

Hlavní motivací byla demonstrace převodu specifických problémů na formát vhodný pro řešiče SAT a možnosti zobrazení řešení z výstupu těchto řešičů. Server dovede zobrazit ke zvolené úloze formuli výrokové logiky v čitelné podobě a rovněž poskytuje možnost stáhnout textový soubor s formulí ve formátu vhodném do řešiče SAT (aktuálně ve formátu DIMACS CNF). Výstupy samozřejmě odpovídají zadaným vstupům uživatele.

V kapitole 2 budou představeny řešiče SAT formou vhodnou i pro osoby, které s nimi doposud nepřišly do styku. Dojde k definici pojmu konjunktivní normální forma, bude zmíněn princip převodu problému na SAT problém, uvedeme některé nástroje pro vyhodnocení splnitelnosti formule v KNF a vysvětlíme základní syntaxi a sémantiku souborů sloužících jako vstup a výstup řešičů SAT ve formátu DIMACS CNF.

Kapitola 3 pojednává o uvažovaných úlohách a jejich převodu na SAT. Mezi řešenými úlohami nalezneme sudoku v sekci 3.1, úplné párování grafu v sekci 3.2 a součet podmnožiny v sekci 3.3. Jednotlivé problémy si vždy nejdříve popíšeme, poté objasníme způsob převodu a na závěr zmíníme vlastnosti typické pro konkrétní problém.

Popisem vývoje výukového serveru se zabývá kapitola 4 a to z pohledu použitých funkcí, technologií, návrhu uživatelského rozhraní atd.

V závěrečné kapitole 5 jsou nastíněny další možnosti rozšíření serveru.

2 Řešiče SAT

Problém splnitelnosti booleovských formulí se v informatice nazývá problém zjišťování, zda existuje nějaké ohodnocení, ve kterém je vstupní formule výrokové logiky splnitelná. Splnitelná formule (satisfiable nebo zkráceně SAT) je taková, k níž existuje valuace přiřazující proměnným hodnoty pravda `true` nebo nepravda `false` tak, aby výsledná formule byla ohodnocena jako `true`. Jestliže žádné takové ohodnocení neexistuje, říkáme, že formule je nesplnitelná (unsatisfiable nebo zkráceně UNSAT). Například formule $a \wedge \neg b$ je splnitelná, jelikož při ohodnocení $a = \texttt{true}$ a $b = \texttt{false}$ je i výsledná formule pravdivá. Naproti tomu formule $a \wedge \neg a$ je nesplnitelná.

SAT je jedním z prvních problémů, u kterého byla dokázána NP-úplnost[1]. Velké množství vědeckých prací bylo věnováno hledání algoritmu, který by efektivně řešil všechny instance problému SAT v polynomiálním čase. Doposud však nebyl nalezen, a ačkoliv se věří, že takový algoritmus neexistuje, nepodařilo se to prozatím matematicky dokázat. Jedná se o jeden z nejznámějších otevřených problémů ve výpočetní teorii. Mnoho vědeckých prací je proto věnováno hledání algoritmu pro řešení SATu, který bude polynomiální pro co největší množství vstupních formulí.

Řešič SAT je program, jenž pro vstupní formuli, zadanou v určitém formátu, dovede zjistit, zda je splnitelná. Je-li formule splnitelná, většina řešičů zobrazí i nějaké ohodnocení proměnných, při kterém je pravdivá. Kromě toho jsou často na výstupu zobrazeny i statistické informace o délce výpočtu, množství použité paměti, hloubce rekurze atd. Některé řešiče dovedou pro nesplnitelné formule najít maximální počet klauzulí, které při nějakém ohodnocení mohou být splněny současně.

Díky velkému zájmu o tuto problematiku vzniká mnoho řešičů s unikátním způsobem výpočtu, které proti sobě mimo jiné soupeří v mezinárodních soutěžích. Na soutěžích se hodnotí, jaký algoritmus je schopný nejrychleji rozhodnout splnitelnost pořadateli zvolené množiny formulí. Pro zjednodušení úsilí vynaloženého při testování a porovnávání algoritmů či heuristik soutěžících řešíčů, bylo potřeba zavést jednotný vstupní formát. Dnešní řešiče SAT jsou natolik sofistikované, že jen pro málo formulí vyžadují exponenciální čas výpočtu, a proto dává smysl je využít i pro řešení některých polynomiálních problémů, jako je například v této práci uvedený problém párování grafu.

Za referenční řešič byl v rámci této práce na základě konzultace zvolen Minisat, jenž vyvinuli Niklas Eén a Niklas Sörensson. Minisat verze 1.13 se v roce 2005 umístil v soutěži SAT competition[2] na druhém místě, čímž dokázal výborné vlastnosti. Navíc je výhodou jeho volná dostupnost. Jedná se o malý (první verze má přibližně 600 řádků kódu, komentáře a prázdné řádky nepočítaje) a přitom efektivní řešič s dobrou dokumentací, jenž obsahuje všechny rysy moderních řešičů. Výstup dovede uložit do souboru a proto se hodí pro účely v této práci popisovaného serveru.

2.1 Specifikace řešiče Minisat

Zdrojové kódy řešiče jsou k dispozici na stránkách autorů, stejně jako předkompilované binární soubory pro Unix i Windows[3]. Varianta pro Windows využívá aplikaci Cygwin[4], což je široká kolekce GNU a Open Source nástrojů, které poskytují funkcio-

nalitu podobnou Linuxové distribuci v prostředí Windows. Alternativně je možné použít některý z online nástrojů, například řešení Minisat v Javascriptu[5].

Referenční řešič Minisat i mnoho dalších používají formát DIMACS CNF (the center for DIscrete MAthematics and theoretical Computer Science, Conjunctive Normal Form). Jedná se o jednoduchý textový formát, jenž reprezentuje formuli v konjunktivní normální formě (KNF). Formule je tedy tvořena následujícími částmi:

- *literál*: booleovská proměnná (např. x_4) nebo její negace ($\neg x_4$), v Minisatu vyjádřena jako kladné číslo 4 respektive -4 pro negaci,
- *klauzule*: disjunkce jednoho nebo více literálů, opakující se literály je možné v klauzuli zredukovat při zachování pravdivostní hodnoty,
- *formule*: konjunkce jedné nebo více klauzulí.

Správně utvořená formule může vypadat například takto: $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (\neg x_3 \vee x_4)$. Libovolná formule může být převedena do KNF, popis algoritmu není předmětem této práce, ale lze jej dohledat například v učebním textu logiky pro informatiky[6].

2.1.1 Vstupní formát

V této sekci bude popsána specifikace souboru ve formátu DIMACS CNF vhodném pro vložení do řešiče Minisat.

Řádek s komentářem je v textu uveden znakem `c`. První řádek, který není komentář, se bude řešič snažit rozpoznat jako preambuli a ta musí mít následující tvar:

```
p cnf POCET_PROMENNYCH POCET_KLAUZULI
```

Každý další řádek, který není komentář, definuje samostatnou klauzuli. Na takovém řádku se nacházejí mezerou oddělené proměnné, jež mají v souboru pouze číselnou hodnotu (např. `1 2 -5 0`). Pozitivní literál, například proměnnou x_5, reprezentuje v souboru pro SAT kladné celé číslo 5. Záporná čísla jsou chápána jako negace proměnných, tedy literál $\neg x_5$ může být reprezentován záporným celým číslem -5. Každý řádek je ukončen znakem `0`, který je od poslední proměnné oddělen mezerou. Začátek vstupu může vypadat například takto:

```
c ARUO 17.11.2014 11:24:01
c Problem: Sudoku
p cnf 9 91
1 2 3 4 5 6 7 8 9 0
−1 −2 0
−1 −3 0
−1 −4 0
. . .
```

V tomto příkladu je na řádku začínajícím znakem `p` specifikován formát souboru (cnf), číslo 9 označuje počet proměnných a číslo 91 počet klauzulí. Každý další řádek pak zastupuje vždy jednu klauzuli.

2.1.2 Spuštění Minisatu

Máme-li v systému integrovaný příslušný soubor s Minisatem, spustíme program příkazem

```
minisat in.txt
```

v příkazové řádce, kde `in.txt` reprezentuje název souboru se vstupem. Tehdy budou v konzoli zobrazeny statistiky vykonaného programu a informace, zda je formule splnitelná nebo ne. Chceme-li získat v případě splnitelné formule i její pravdivostní ohodnocení, musíme vypsat výstup do souboru tak, že zadáme jeho název jako druhý parametr. Příkaz pak může vypadat například takto:

```
minisat in.txt out.txt
```

2.1.3 Výstupní formát

Po spuštění Minisatu se na obrazovku vypíše kromě statistiky o běhu programu i slovo `SATISFIABLE` nebo `UNSATISFIABLE` v závislosti na tom, zda je formule splnitelná či nikoliv. Zvolíme-li vypsání výstupu do souboru, na první řádek se vypíše slovo `SAT` (je-li formule splnitelná) nebo `UNSAT` (není-li formule splnitelná). V případě, že je formule splnitelná, následuje na druhém řádku výčet proměnných ve valuaci, která zajistila platnost formule. V tomto výčtu budou kladná čísla zastupovat proměnné ohodnocené jako `true` a záporná čísla proměnné ohodnocené jako `false`. Existuje-li více možných řešení, vypíše řešič první nalezené. Výstup může vypadat například takto:

```
SAT
1 -2 -3
```

Dostaneme-li tento výstup, víme, že formule je pravdivá při ohodnocení $x_1 = \texttt{true}$, $x_2 = \texttt{false}$ a $x_3 = \texttt{false}$.

Popis formátu byl inspirován uživatelským manuálem pro Minisat, který napsal David A. Wheeler[7].

3 Uvažované problémy a jejich převod na SAT

V této kapitole budou rozebrány jednotlivé řešené problémy. Po stručném obecném úvodu následuje pro každý problém popis zvoleného převodu na SAT.

Obecně je cílem převodu mezi dvěma rozhodovacími problémy nalezení algoritmu, který převede vstup zdrojového problému na vstup cílového problému tak, aby odpověď typu ano / ne zůstala stejná (tedy je-li odpověď ano, zůstává ano). Nejde-li o rozhodovací problémy, potřebujeme rovněž najít algoritmus pro převod výstupu cílového problému na výstup zdrojového problému.

V případě, kdy je cílovým problémem SAT, je potřeba najít vhodný algoritmus pro převod konkrétního problému na vstupní formuli ve formátu vhodném pro SAT řešiče a také pro převod výsledného ohodnocení proměnných získaného z řešiče zpět na formát zdrojového problému. Může existovat více různých algoritmů pro převod konkrétního problému na problém SAT lišících se časovou složitostí, velikostí výsledné formule, atd. Konkrétní algoritmy pro tři řešené problémy byly zvoleny po konzultaci s vyučujícím předmětu ARUO tak, aby každý demonstroval specifické myšlenky často používané v převodech na SAT.

3.1 Sudoku

Sudoku je matematický hlavolam, který vymyslel v roce 1979 Howard Garns pod názvem „Number place“. Hra se těší velké oblibě v Japonsku, kde byl vymyšlen dnes po celém světě používaný název sudoku.

Hrací pole sudoku sestává z mřížky 9×9, do které vyplňujeme čísla od jedné do devíti tak, aby v každém řádku, sloupci i čtverci o rozměru 3×3 (kterých je na hracím poli devět) bylo každé číslo pouze jednou. Některá políčka jsou již předvyplněna a cílem je doplnit celou mřížku tak, aby byly zachovány zmíněné podmínky. Obecně platí, že čím méně čísel je předvyplněno, tím je obtížnost hlavolamu větší (ale ne vždy je to pravidlem). Hra nevyžaduje žádné speciální znalosti nebo matematické výpočty. Člověk si vystačí s vlastní hlavou a trochou koncentrace. Hra sudoku se dočkala mnoha různých variant, těmi se v této práci však nebudeme zabývat.

3.1.1 Převod sudoku na SAT

Hrací pole představuje z pohledu formule $9 \times 9 \times 9$ proměnných x_{ijk}, kde i představuje řádek (počítáno z vrchu od hodnoty 1), j indikuje sloupec (počítáno zleva od hodnoty 1) a k zastupuje konkrétní číslo, jež v případě, že je proměnná x_{ijk} ohodnocena jako pravda (`true`), na tomto místě skutečně leží.

Řešení je vyjádřeno jako formule v konjuktivní normální formě. Pro zjednodušení si ji můžeme představit jako konjunkci pěti podformulí, přičemž každá je také v KNF.

$$\varphi = \varphi_{tiles} \wedge \varphi_{rows} \wedge \varphi_{columns} \wedge \varphi_{squares} \wedge \varphi_{assignment}$$

Podformule φ_{tiles} specifikuje, že každému poli v mřížce je řešením přiděleno právě jedno číslo, což je vyjádřeno konjunkcí 9×9 formulí $\varphi_{tile}^{i,j}$. Každá z těchto formulí zajišťuje,

aby políčko na i-tém řádku a v j-tém sloupci mělo jedno z 9-ti možných čísel a rovněž, aby pro každou dvojici různých čísel vždy alespoň jedno z čísel nebylo v tomto políčku.

$$\varphi_{tiles} = \bigwedge_{i=1}^{9} \bigwedge_{j=1}^{9} \varphi_{tile}^{i,j}$$

$$\varphi_{tile}^{i,j} = (x_{ij1} \vee x_{ij2} \vee x_{ij3} \vee x_{ij4} \vee x_{ij5} \vee x_{ij6} \vee x_{ij7} \vee x_{ij8} \vee x_{ij9}) \wedge \bigwedge_{m=1}^{9} \bigwedge_{n=m+1}^{9} (\neg x_{ijm} \vee \neg x_{ijn})$$

φ_{rows} vyjadřuje pravidlo, že každý řádek musí mít každé číslo z 9-ti možných obsaženo právě jednou. To je vyjádřeno konjunkcí 9×9 podformulí $\varphi_{row}^{i,k}$, ve které každá podformule $\varphi_{row}^{i,k}$ v první klauzuli zajišťuje, aby na i-tém řádku bylo číslo k alespoň jednou a zbylé klauzule zajišťují, aby pro každou dvojici různých polí na řádku i vždy alespoň jedno z nich neobsahovalo číslo k.

$$\varphi_{rows} = \bigwedge_{i=1}^{9} \bigwedge_{k=1}^{9} \varphi_{row}^{i,k}$$

$$\varphi_{row}^{i,k} = (x_{i1k} \vee x_{i2k} \vee x_{i3k} \vee x_{i4k} \vee x_{i5k} \vee x_{i6k} \vee x_{i7k} \vee x_{i8k} \vee x_{i9k}) \wedge \bigwedge_{m=1}^{9} \bigwedge_{n=m+1}^{9} (\neg x_{imk} \vee \neg x_{ink})$$

$\varphi_{columns}$ obdobným způsobem vyjadřuje pravidlo, že v každém sloupci musí být každé číslo právě jednou.

$$\varphi_{columns} = \bigwedge_{j=1}^{9} \bigwedge_{k=1}^{9} \varphi_{column}^{j,k}$$

$$\varphi_{column}^{j,k} = (x_{1jk} \vee x_{2jk} \vee x_{3jk} \vee x_{4jk} \vee x_{5jk} \vee x_{6jk} \vee x_{7jk} \vee x_{8jk} \vee x_{9jk}) \wedge \bigwedge_{m=1}^{9} \bigwedge_{n=m+1}^{9} (\neg x_{mjk} \vee \neg x_{njk})$$

Rovněž podformule $\varphi_{squares}$ používá obdobný způsob vyjádření pravidla. Každý čtverec smí obsahovat všechna možná čísla právě jednou. Jednotlivé čtverce 3×3 začínají na následujících souřadnicích: $X = \{(1,1), (1,4), (1,7), (4,1), (4,4), (4,7), (7,1), (7,4), (7,7)\}$.

$$\varphi_{squares} = \bigwedge_{x \in X} \bigwedge_{k=1}^{9} \varphi_{square}^{x,k}$$

$$\begin{aligned} \varphi_{square}^{(i,j),k} = (&x_{ijk} \vee x_{i(j+1)k} \vee x_{i(j+2)k} \vee x_{(i+1)jk} \vee x_{(i+1)(j+1)k} \vee \\ &x_{(i+1)(j+2)k} \vee x_{(i+2)jk} \vee x_{(i+2)(j+1)k} \vee x_{(i+2)(j+2)k}) \wedge \\ &\bigwedge_{m=0}^{8} \bigwedge_{n=m+1}^{8} (\neg x_{(\lfloor (m/3) \rfloor + i + 1)((m\%3)+j+1)k} \vee \\ &\neg x_{(\lfloor (n/3) \rfloor + i + 1)((n\%3)+j+1)k}) \end{aligned}$$

V neposlední řadě podformule $\varphi_{assignment}$ vyjadřuje předvyplněné hodnoty konkrétního zadání sudoku, tedy hodnoty na serveru zadané uživatelem. Například číslo 6 na pátém řádku a třetím sloupci je vyjádřeno klauzulí x_{536}.

$$\varphi_{assignment} = \bigwedge_{(i,j,k)\in Y} x_{ijk}$$

kde $Y = \{(i,j,k) \mid k \text{ je číslo na souřadnicích } (i,j)\}$

Jedině podformule $\varphi_{assignment}$ je závislá na uživatelském vstupu a může se měnit zadání od zadání. Ostatní podformule jsou pro všechna zadání stejné. Existují další způsoby, jak vytvořit formuli pro sudoku. Některé generují méně klauzulí, ale je komplikovanější je pochopit, vyjádřit a ověřit jejich správnost.

3.2 Úplné párování grafu

Celá řada v literatuře definovaných problémů vychází z teorie grafů. Než si popíšeme konkrétní problém, připomeňme si základní definici párování grafu.

Nechť graf $G = (V, E)$ je neorientovaný graf. Množina $M \subseteq E$ se nazývá párování grafu G pokud žádné dvě hrany v M nemají společný vrchol. Prázdná množina je párování na každém grafu (nemá žádné páry - hrany). Graf bez hran má pouze prázdné párování. Úplné párování grafu je takové, ve kterém každý vrchol má právě jednu svou incidentní hranu[8].

V rámci této práce je řešen problém hledání úplného párování grafu.

3.2.1 Převod párování grafu na SAT

Označme si vrcholy grafu $A_1, A_2, \ldots, A_n$. Nechť $e(A_i)$ je množina všech hran přímo spojených s vrcholem A_i. Hrana mezi vrcholy A_i a A_j je pro usnadnění matematického zápisu vyjádřena jako množina se dvěma vrcholy $\{A_i, A_j\}$, jelikož uvažujeme neorientované hrany grafu, ve kterých nezáleží na pořadí vrcholů.

Pro každou hranu $\{A_i, A_j\}$ je vytvořena proměnná $x_{A_i-A_j}$, jež může být označena rovněž jako $x_{A_j-A_i}$. Jestliže proměnná $x_{A_i-A_j}$ je ohodnocena jako pravda (`true`), zahrneme hranu $\{A_i, A_j\}$ do párování. V opačném případě je-li proměnná ohodnocena jako nepravda (`false`), hranu do párování nezahrneme.

Pravidlo zajišťující, aby žádné dvě hrany neměly společný vrchol znázorňuje následující formule:

$$\varphi = \bigwedge_{i=1}^{n} \varphi_{A_i}$$

Každá podformule φ_{A_i} zajišťuje potřebná pravidla pro jeden vrchol:

$$\varphi_{A_i} = \left(\bigvee_{\{A_i,A_j\}\in e(A_i)} x_{A_i-A_j} \right) \wedge \bigwedge_{\substack{\{A_i,A_j\}\in e(A_i),\\ \{A_i,A_k\}\in e(A_i),\\ A_j\neq A_k}} (\neg x_{A_i-A_j} \vee \neg x_{A_i-A_k})$$

Díky první klauzuli zajistíme zahrnutí alespoň jedné hrany v párování. Ostatní klauzule zajišťují, aby nedošlo k zahrnutí dvou různých hran současně.

3.3 Součet podmnožiny

Problém součtu podmnožiny a jeho varianta dělení kořisti (známá také jako problém „loupežníka“) jsou NP-úplné kombinatorické úlohy. To znamená, že existuje nedeterministický Turingův stroj, který v polynomiálním čase zjistí, zda existuje řešení[9] a předpokládá se, že neexistuje deterministický Turingův stroj, který by dokázal totéž.

3.3.1 Převod součtu podmnožiny na SAT

Instance problému je dána k přirozenými čísly $N_0, \ldots, N_{k-1}$ a požadovaným součtem S. V případě varianty dělení kořisti je namísto požadovaného součtu brána polovina součtu všech vstupujících čísel. Pro zjednodušení použijeme stejné značení S.

Mějme proměnné $n_0, n_1, \ldots, n_{k-1}$, kde každá reprezentuje číslo se stejným indexem a určuje, zda je číslo součástí podmnožiny, která tvoří řešení.

Při sčítání můžeme u konkrétního čísla N_i ponechat bity s hodnotou 0 nezměněny a bity s hodnotou 1 nahradit pomocnou proměnnou n_i. Například číslo $N_2 = 25$ je pak reprezentováno jako $n_2 n_2 00 n_2$. Pokud bude toto číslo zahrnuto v podmnožině tvořící řešení, tedy proměnná je ohodnocena jako pravda ($n_2 = 1$), přičítáme číslo v jeho vlastní hodnotě $11001_{bin} = 25_{dec}$. Naopak, není-li číslo zahrnuto v řešení, je v součtu nahrazeno reprezentací 00000, jež řešení dále neovlivňuje.

Nechť m je počet bitů potřebných k reprezentaci maximálního čísla z množiny:

$$\{S, N_0, N_1, \ldots, N_{k-1}\}$$

Řešení se inspiruje maticí $k \times m$ bitových sčítaček znázorněných na obrázku 1.

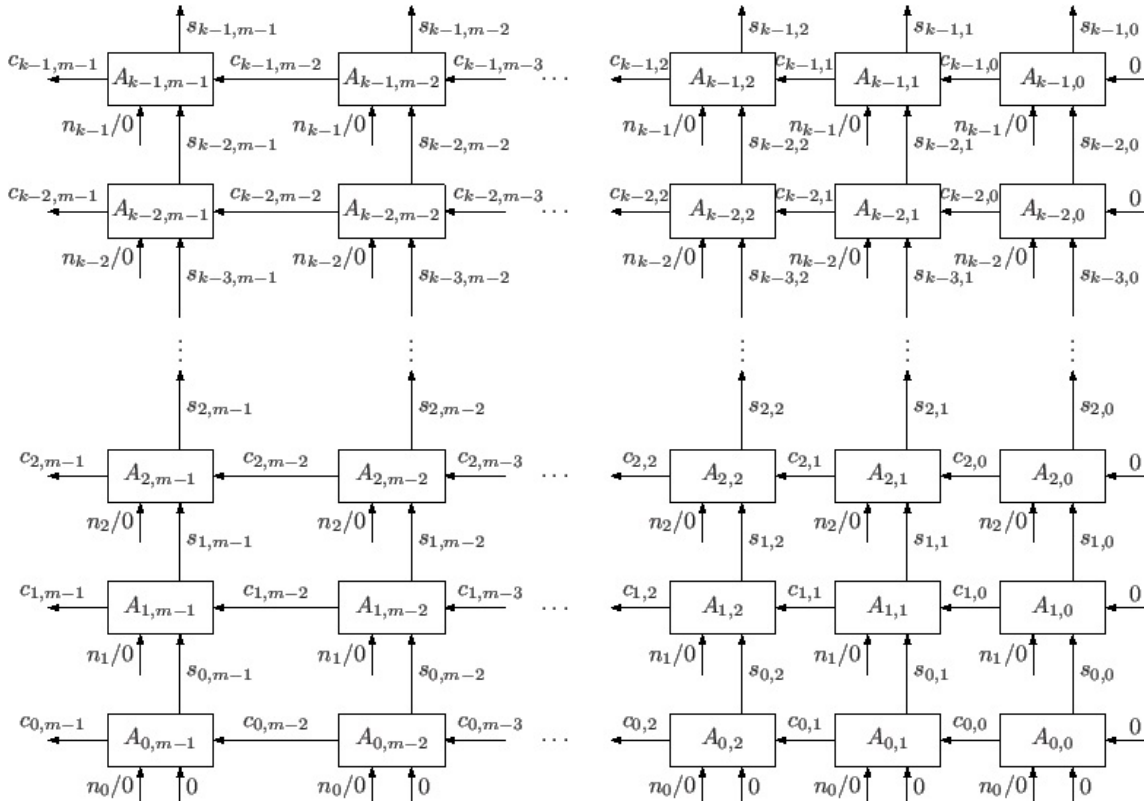
Na i-tém řádku je číslo N_i přičteno k součtu předchozích čísel. Vstup $n_i/0$ znázorňuje dříve zmiňovanou reprezentaci čísla. Tedy, že na vstupu se nachází n_i nebo 0 dle konkrétního bitu čísla. Nejméně významné bity jsou sčítány ve sloupci nejvíce vpravo. Nejvýznamnější bity jsou sčítány ve sloupci nejvíce vlevo.

V matici se vyskytuje $k \times m$ proměnných $s_{i,j}$ a $k \times m$ proměnných $c_{i,j}$. Každá proměnná $s_{i,j}$ reprezentuje bit součtu zvolených čísel z množiny $\{N_0, \ldots, N_i\}$ na pozici j (počínaje nulou počítáno od nejméně významného bitu). Každá proměnná $c_{i,j}$ reprezentuje přenos do vyššího řádu po součtu bitu na pozici j čísla N_i s bitem na pozici j součtu zvolených čísel z množiny $\{N_0, \ldots, N_{i-1}\}$. V prvním řádku jsou na vstupu místo součtu z předchozího řádku nulové bity. Obdobně namísto přenosu z méně významného bitu jsou ve sloupci nejvíce vpravo na vstupu nulové bity.

Pro řešič SAT je použita následující formule:

$$\varphi = \varphi_{Sum} \wedge \varphi_{Carry} \wedge \bigwedge_{i=0}^{k-1} \bigwedge_{j=0}^{m-1} \varphi_{Adder}^{i,j}$$

Nyní se pokusíme popsat všechny tři části formule.



Obrázek 1: Matice bitových sčítaček

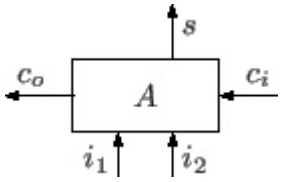
$$\varphi_{Sum} = \bigwedge_{j=0}^{m-1} g_i s_{k-1,j}$$

kde g_i je prázdná, pokud bit na pozici i součtu S je 1. V případě, že bit na pozici i součtu S je 0, pak g_i reprezentuje unární operátor negace $\neg$. Tato formule zajišťuje, aby součet vybraných čísel byl přesně roven součtu S. Například pro $S = 23_{dec} = 10111_{bin}$ by formule vypadala takto: $s_4 \wedge \neg s_3 \wedge s_2 \wedge s_1 \wedge s_0$.

$$\varphi_{Carry} = \bigwedge_{i=0}^{k-1} \neg c_{i,m-1}$$

pomocí této podformule zajišťujeme, že z nejvíce významných bitů již nejsou žádné přenosy. V některých případech součet čísel může mít více než m bitů, ale v takovém případě by byl součet větší než S a tudíž bychom nezískali platné řešení. Všechna správná řešení mají nanejvýš m bitů a tudíž jsou v souladu s podformulí φ_{Carry}.

Každá formule $\varphi^{i,j}_{Adder}$ se stará, aby oba výstupy z konkrétní bitové sčítačky byly korektní vůči vstupům. Pro názornost uvažujme jednu bitovou sčítačku dle následujícího obrázku:



Obrázek 2: Bitová sčítačka

Formule φ_{Adder} pro tuto bitovou sčítačku vypadá následovně:

$$\begin{aligned}\varphi_{Adder} = &(i_1 \vee i_2 \vee c_i \vee \neg s) \wedge (i_1 \vee i_2 \vee \neg c_i \vee s) \\ &\wedge (i_1 \vee \neg i_2 \vee c_i \vee s) \wedge (i_1 \vee \neg i_2 \vee \neg c_i \vee \neg s) \\ &\wedge (\neg i_1 \vee i_2 \vee c_i \vee s) \wedge (\neg i_1 \vee i_2 \vee \neg c_i \vee \neg s) \\ &\wedge (\neg i_1 \vee \neg i_2 \vee c_i \vee \neg s) \wedge (\neg i_1 \vee \neg i_2 \vee \neg c_i \vee s) \\ &\wedge (i_1 \vee c_i \vee \neg c_o) \wedge (i_1 \vee i_2 \vee \neg c_o) \wedge (i_2 \vee c_i \vee \neg c_o) \\ &\wedge (\neg i_1 \vee \neg i_2 \vee c_o) \wedge (\neg i_1 \vee \neg c_i \vee c_o) \wedge (\neg i_2 \vee \neg c_i \vee c_o)\end{aligned}$$

V prvních osmi klauzulích zajišťujeme, že proměnné součtu s je nastavena správná hodnota v závislosti na vstupních proměnných. Dalších šest klauzulí zajišťuje nastavení správné hodnoty přenosu do vyššího řádu v závislosti na vstupních proměnných. Například je-li $i_1 = 1, i_2 = 1$ a $c_i = 1$, oba výstupy (součet i přenos) jsou rovny 1. V případě, že by součet byl $s = 0$, klauzule $(\neg i_1 \vee \neg i_2 \vee \neg c_i \vee s)$ je nepravdivá (a tudíž celá formule bude nepravdivá), obdobně je-li přenos do vyššího řádu $c_o = 0$, pak klauzule $(\neg i_1 \vee \neg c_i \vee c_o)$, $(\neg i_1 \vee \neg i_2 \vee c_o)$ a $(\neg i_2 \vee \neg c_i \vee c_o)$ jsou nepravdivé.

Podformule $\varphi^{i,j}_{Adder}$ pro každou bitovou sčítačku $A_{i,j}$ je vytvořena podle této šablony dosazením konkrétních proměnných bitové sčítačky $A_{i,j}$ dle matice všech bitových sčítaček zobrazené na obrázku 1 místo proměnných i_1, i_2, c_i, c_o, s.

Pro zjednodušení generované formule, je-li na vstupu nula 0 (přenos vstupující do sloupce nejvíce vpravo, součet vstupující do spodního řádku nebo nulové bity konkrétního vstupního čísla ze zadání) dochází k redukci podformule. Má-li být 0 dosazena na místo pozitivního literálu v klauzuli, je rovnou vynechána, jelikož nemá vliv na výsledné ohodnocení klauzule. Má-li být 0 dosazena namísto negované proměnné, dochází k vypuštění celé klauzule, jelikož v tom případě při libovolném ohodnocení ostatních proměnných jde vždy o pravdivou klauzuli. Tímto vypouštěním proměnných může dojít k výskytu duplicitních klauzulí, když jsou vypuštěny právě literály, na nichž se klauzule lišily. V současné době server neprovádí žádnou kontrolu na výskyt duplicitních klauzulí.

4 Vývoj výukového serveru

Na počátku byla myšlenka webového serveru určeného pro studenty předmětu ARUO, jež by názorně ukázal možné postupy používané při převodu problémů různých typů na formuli výrokové logiky v KNF a pomocí popisu teorie i názorných příkladů vysvětlil funkci jednotlivých klauzulí.

Uživatel by tedy měl mít možnost zadat pro zvolený problém vstupní parametry, zobrazit si formuli, případně její část a získat soubor ve formátu vhodném pro vložení do řešiče SAT. Ohodnocení formule v řešiči provádí uživatel svépomocí bez využití serveru. Výstup z řešiče, tedy informaci o splnitelnosti formule a v kladném případě i nalezené platné ohodnocení, by pak mělo být možné na server nahrát pro zobrazení řešení.

Kromě ovládacích funkcí uživatele bude na webu k dispozici i teoretické vysvětlení částí formule. V rámci této práce bude implementováno rozhraní ke třem problémům: Sudoku 3.1, Úplné párování grafu 3.2 a Součet podmnožiny 3.3.

4.1 Funkční specifikace

Z pohledu uživatele jsou pro jednotlivé problémy k dispozici tyto funkce:

1. Formulář pro zadání uživatelského vstupu a zobrazení řešení problému,
2. sekce pro manipulaci s uživatelským vstupem:
   (a) uložení vstupu do paměti
   (b) stažení textového souboru s uživatelským vstupem
   (c) nahrání textového souboru s uživatelským vstupem
3. sekce pro manipulaci s generovanou formulí,
   (a) stažení formule ve formátu DIMACS CNF
   (b) vypsání formule nebo její části na obrazovku
   (c) nahrání výstupu z řešiče SAT pro evaluaci a zobrazení výsledku

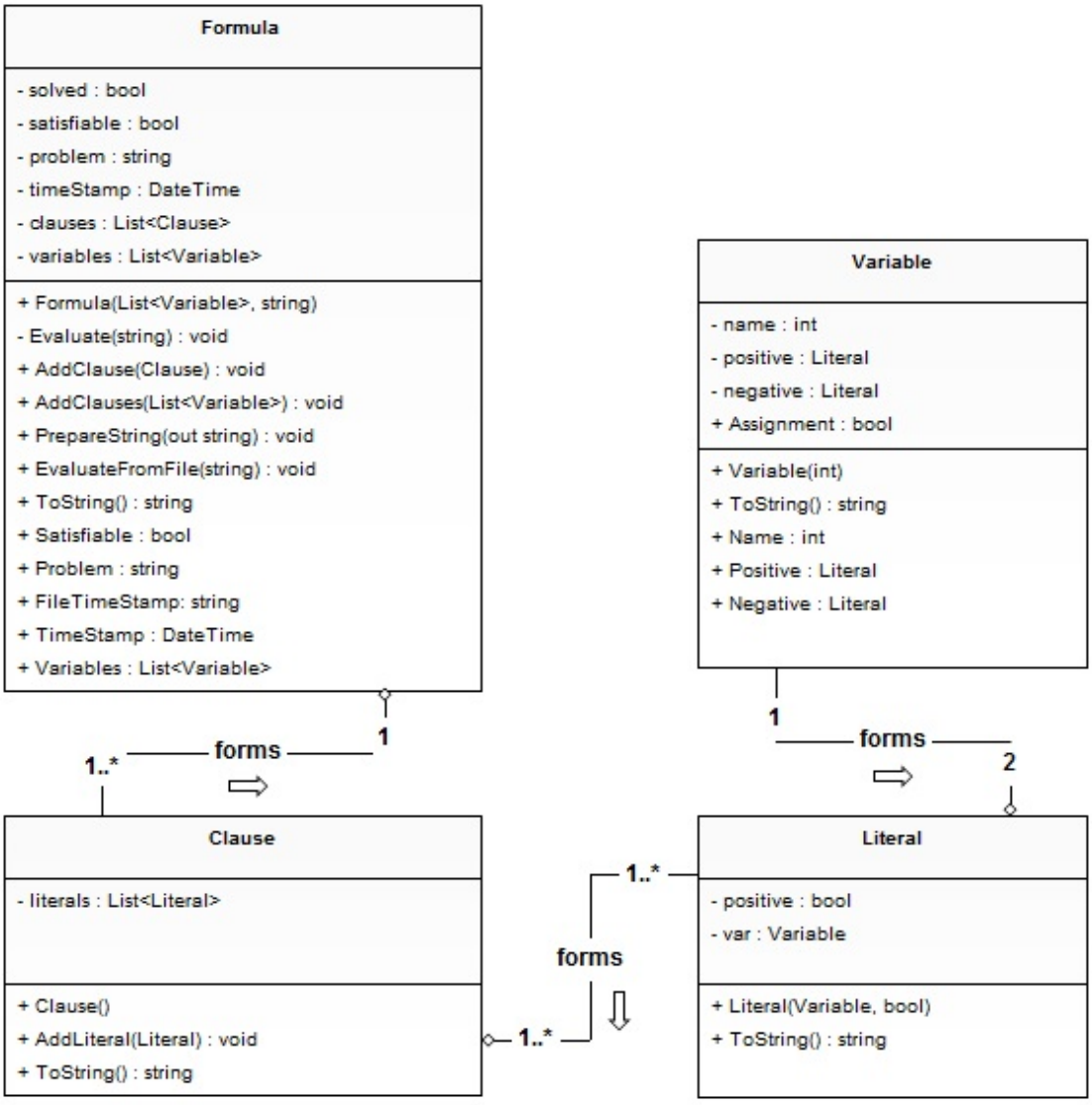
Detailní popis ovládání je k dispozici v uživatelské příručce, jež je přiložena na CD.

4.2 Technická specifikace

Jelikož se jedná o implementaci serveru dostupného v síti internet, bylo potřeba zvolit vhodnou platformu s podporou jazyka HTML. Vybrán byl ASP.NET framework, open source řešení webové aplikace běžící na serveru[10], jenž poskytuje vhodné prostředí Web Forms pro tvorbu dynamických HTML formulářů. Prostředí je založeno na použití ovládacích prvků řízených událostmi, jejichž vykonávání lze ponechat zcela v režii serveru[11]. Pro implementaci byl zvolen jazyk C#.

Stěžejní částí aplikace jsou třídy reprezentující formuli, jejich třídní diagram je znázorněn na obrázku 3. Jedná se o čtyři třídy:

- proměnná `Variable`,
- literál `Literal` - proměnná nebo její negace,
- klauzule `Clause` - disjunkce proměnných a
- formule `Formula` - konjunkce klauzulí.



Obrázek 3: Třídní diagram 1: Reprezentace formule

Implementované problémy jsou reprezentovány samostatnými třídami (např. třída `Sudoku` u problému téhož jména). Společné vlastnosti a metody se dědí z rodičovské

třídy `Problem`. Některé problémy vyžadují další třídy pro specifické objekty. Součet podmnožiny pro své účely využívá navíc třídu `Number` reprezentující 64 bitový integer jako číslo i jako pole bitů. Úplné párování grafu vyžaduje reprezentaci grafu v paměti a pro tyto účely byly zkonstruovány třídy:

- graf `Graph`,
- vrchol `Vertex` a
- hrana `Edge`.

Třídy reprezentující problémy a jejich vazby jsou znázorněny na obrázcích 4 a 5. Nutno podotknout, že třídní diagramy byly tvořeny již pro konkrétní platformu a pozornému oku neunikne coding style a datové typy typické pro jazyk C#.

Kromě tříd reprezentujících problémy bylo potřeba implementovat uživatelské rozhraní. Pro tento účel bylo využito návrhového vzoru model-view-presenter (MVP) ve variantě Passive View. Tento vzor je specifický tím, že mezi doménovou logikou (model) a prezentační vrstvou (view) není žádná vazba. View zůstává naprosto pasivní a veškerou logiku (reakce na změny vyvolané uživatelem, načítání hodnot pro zobrazení) přenechává kontroloru (controller)[12]. Mezi těmito dvěma vrstvami je použit vzor observer, tedy view registruje controller v kolekci observerů a v případě, že dojde ke změně, jsou jednotlivé záznamy z kolekce informovány[13]. Díky tomuto vzoru dochází ke snížení vazeb mezi vrstvami.

4.3 Podpora pro implementaci dalších problémů

Pro další rozšíření je možno využít obecně psaných částí aplikace. Ve jmenném prostoru `Model` se jedná o třídy reprezentující:

1. problém, formuli a její části
2. graf a jeho části u problémů z teorie grafů

Díky těmto třídám již není potřeba znovu řešit generování souboru v DIMACS CNF, načítání ohodnocení proměnných formule ze souboru s výstupem z řešiče SAT a ukládání grafu do textového souboru a jeho načítání.

Pro logiku dalšího problému vytvoříme samostatnou složku, do které ukládáme třídy specifické pro konkrétní typ problému. Obecné třídy ukládáme tak, aby byly případně dostupné pro nové problémy ze stejného oboru (například problémy z teorie grafů jsou ukládány do složky `Graphs` včetně pomocných tříd reprezentujících vrchol či hranu).

Některé problémy mají natolik specifické vlastnosti, že pro ně můžeme vytvořit samostatnou složku (například problém sudoku).

Z pohledu uživatelského rozhraní se pro každý nový problém vytvoří samostatná stránka typu Web Forms ve složce View. Součástí stránky by v tom případě měl být i soubor s příponou `.aspx.cs`, jenž musí implementovat rozhraní `Views.cs` pro zachování návrhového vzoru, který byl popsán dříve v této kapitole. Veškerá logika by měla být

vykonávána pomocí controlleru. Všechny controllery se nachází ve složce `Controller` a je pro ně specifické, že jsou potomky třídy `Controller.cs`, ve které se mimo jiné nachází hlavní mechanismus pro výběr správné reakce na akce uživatele. V současnosti jsou controllery vyvinuty dle vzoru transakčního skriptu (pro každý případ užití existuje samostatná metoda), proto je funkcionalita samostatně pro jednotlivé problémy. V rámci dalšího vývoje a udržitelnosti zdrojového kódu by mohlo dojít k refaktoringu na vhodnější návrhový vzor (např. doménový model).

Pro zobrazování formule pro konkrétní uživatelské vstupy by také mohla být podpora ve třídě `Formula`, ale pro většinu problémů bývá formule příliš velká a je vhodnější zobrazovat pouze její jednotlivé podformule. Rozdělení na podformule je dáno specifickými vlastnostmi jednotlivých problémů a proto je pro každý problém toto zobrazování řešeno samostatně.

4.4 Použité technologie

Webová prezentace je psána v jazyce HTML5. Pro zpříjemnění uživatelského dojmu jsou použity základní kaskádové styly CSS. Drobné pomocné skripty jsou napsány v jazyce JavaScript. Tyto tři technologie jsou spolu kompatibilní díky dlouholeté snaze mezinárodní komunity vývojářů W3C. Kromě vlastních uživatelských skriptů je použita knihovna jQuery pro usnadnění manipulace s elementy stránek. Pro zobrazení teorie k jednotlivým problémům je použita jQuery knihovna colorbox, kterou vyvíjí Jack Moore[14]. Plná funkčnost je tedy zaručená pouze v prohlížeči s povolenými uživatelskými skripty (`JavaScript`).

Popis formule byl vysázen pomocí systému LaTeX, jen graf sčítání (viz obrázek 1) u problému sčítání podmnožiny vyžadoval sofistikovanější zobrazovací mechanismus. Naštěstí existuje LaTeX knihovna GasTeX[15], jež usnadňuje modelování grafů a automatů. Knihovnu vyvinul Paul Gastin z university ENS de Cachan ve Francii. Převod popisu byl uskutečněn prostřednictvím programu LaTeX2HTML[16]. Tento program se již od roku 2001 nevyvíjí a používá tudíž zastaralou specifikaci jazyka HTML. Výstup z programu bylo nutno přepsat do vhodného formátu. Jelikož v současné době většina webových prohlížečů nepodporuje sázení matematických textů ve vyšší kvalitě (například s využitím `MathML`, popis formulí byl exportován jako obrázek.

Aplikace běží na platformě ASP.NET, přičemž je napsána v jazyce C#. Zkušební verze serveru se nachází na adrese `http://aruo-vsb.aspone.cz`, subdoménu a webhosting zdarma poskytla společnost ASPone.

4.5 Nasazení

Aplikaci je možné nasadit na Windows server, který podporuje .NET framework verze 4.5. Na serveru vytvoříme do příslušné složky dle specifikace webhostingu adresářovou strukturu do níž nahrajeme potřebné soubory:

- `bin` - DLL knihovny pro release verzi
- `Content` - obsahuje podsložky:

  - `css` - kaskádové styly
  - `files` - dokumenty
  - `images` - obrázky k problémům a pluginu colorbox
  - `js` - vlastní uživatelské skripty a plugin colorbox
- `temp` - složka se používá při nahrávání souborů uživatelem pro dočasné uložení
- `View` - soubory Win Forms s příponou `.aspx`
- do kořenové složky patří tyto soubory: `Default.aspx`, `Global.asax`, `Layout.Master` a `Web.config`

Celá tato struktura včetně souborů aktuální verze je k dispozici na přiloženém CD v adresáři `application/release/`. Dojde-li při dalším vývoji ke změnám ve zdrojových souborech, je potřeba provést `build` nové verze a změněné soubory nahrát na server.

4.6 Uživatelské rozhraní

Podstránka konkrétního problému je rozdělena do čtyř hlavních sekcí. Rozmístění sekcí je zobrazeno na obrázku 6.

1. Pole pro zadání uživatelského vstupu a zobrazení řešení problému,
2. sekce pro manipulaci s uživatelským vstupem,
3. sekce pro manipulaci s generovanou formulí,
4. místo s teoretickým popisem problému a možností zobrazit formuli nebo její zvolenou část.

Další prvky na stránce (vyznačené na obrázku 6 zelenou barvou):

a. Navigační menu,
b. informace o aktuální `session` a možnost začít `session` novou,
c. stručný návod k řešení problému,
d. místo ke zobrazení hlášení pro uživatele.

4.7 Formát uživatelského vstupu

Implementace jednotlivých problémů vyžadovala vymyšlení způsobu, jak aplikaci předat uživatelský vstup. Původně bylo možné pouze nahrát textový soubor s uživatelským vstupem. V této sekci si popíšeme formát souboru pro jednotlivé problémy. Je nutné zachovat stanovený formát souboru, jinak se může aplikace chovat nekorektně. Rovněž je důležité, aby měl textový soubor nastaveny konce řádků CRLF typické pro Dos/Windows platformu, jinak jej server nebude schopen správně parsovat. Během vývoje došlo

k implementaci formuláře pro zadání uživatelského vstupu. Omezení a způsob validace bude rovněž popsán.

Ukázkové příklady souborů s uživatelskými vstupy jsou k dispozici na přiloženém CD.

4.7.1 Sudoku

Uživatel má k dispozici formulář sestávající z políček 9×9. Do formuláře vpisuje číselné vstupní hodnoty v rozmezí $1 - 9$. Jednotlivá políčka používají HTML5 validaci, jež umožňuje vepsat pouze jeden znak a díky regulárnímu výrazu `[1-9]` musí být tento znak pouze číslo v požadovaném rozmezí. Funkce validace je pro textové pole v jazyce HTML5 zajištěna pomocí atributů `pattern` a `maxlength`:

```
<input type="text" pattern="[1−9]" id="sudokuBox11" size="1" maxlength="1" name="
    ctl00$content$ctl00">
```

Pro uživatele, kteří nechtějí vyplňovat zadání do formuláře, je zde možnost nahrát zadání z textového souboru. Soubor obsahuje 9 řádků, na každém je 9 znaků - čísla a mezery. Nevíme-li, jaké číslo bude na konkrétním umístění, nahradíme jej mezerou. Korektní vstup může vypadat například takto:

```
356  89
1 82   5
4 256 1
 4 6    17
 31    46
76    1 3
   3 268 5
 2    43 1
   43   726
```

4.7.2 Párování grafu

Server v současné době podporuje základní způsob zadávání grafu založený na textové bázi. Uživatel má k dispozici formulář, v němž každý řádek obsahuje dvě pole pro text. Validace je opět zajištěna pomocí HTML5. Menší pole vlevo specifikuje název zdrojového vrcholu. Aby byl zdrojový vrchol zadán ve správném formátu, je u formuláře nastavena validace pomocí regulárního výrazu:

```
[0-9A-Za-z]{1,3}
```

Ten zajišťuje možnost do pole zadat maximálně tři znaky - písmena nebo čísla. Druhé větší pole vpravo slouží pro zadání názvů všech vrcholů, které jsou se zdrojovým propojeny hranou. Jednotlivé vrcholy jsou od sebe odděleny mezerou nebo čárkou. Omezení na tři znaky bylo zvoleno kvůli přehlednosti a pro ilustraci je naprosto postačující, ačkoliv je tím rozsah možných vstupů omezen. O korektní formát zadané množiny cílových vrcholů se opět stará regulární výraz a to v následujícím formátu:

```
([0-9A-Za-z]\{1,3\}[\s,]+)*[0-9A-Za-z]{1,3}
```

Názvy vrcholů mohou být opět maximálně o třech znacích. Validace je v tomto případě zajištěna atributy:

- `required` - pole je povinné a bez jeho vyplnění ve správném formátu není možné formulář odeslat
- `pattern` - obsahuje regulární výraz pro požadovaný formát
- `maxlength` - určuje maximální počet znaků
- `title` - je dobrou praktikou v případě, že je vyplnění pole povinné, uvádět také textový popis požadovaného formátu vstupu[1].

```
<input type="text" required pattern="[0−9A−Za−z]{1,3}" size="3" maxlength="3" id="
    bipartParrent1" name="bipartParrent1" title="Insert␣1␣to␣3␣alphanumeric␣characters" />
<input type="text" required pattern="([0−9A−Za−z]{1,3}[\s,]+)∗[0−9A−Za−z]{1,3}" id="
    bipartChildren1" name="bipartChildren1" title="Insert␣1␣to␣3␣alphanumeric␣characters␣for␣
    each␣vertex.␣Separate␣vertices␣by␣comma␣or␣space." />
```

Další řádky je možné přidávat pomocí tlačítka + a odstraňovat pomocí tlačítka −.

Rovněž pro tento problém je možné nahrát textový soubor se zadáním. Počet řádků v souboru není omezen, struktura řádku však musí zůstat zachována. Zdrojový vrchol je od zbytku řádku oddělen dvojtečkou, mezi cílovými vrcholy musí být vždy mezera. Pro textový soubor není počet znaků jednoho vrcholu omezen. Takové zadání však může způsobit špatnou čitelnost při zobrazení ve webovém prohlížeči. Správnost generované formule bude v každém případě zachována. Samotný textový soubor pak může vypadat například takto:

```
a:k l m
b:k l n
c:k m n o
d:m n o p
e:o p
f:k l p
```

Aplikace provádí základní kontrolu duplicity hran. I když uživatel zadá více než jednu hranu mezi dvěma vrcholy, pro účely zobrazení a generování formule bude vždy použita pouze jedna. Také smyčky u vrcholu nejsou brány do úvahy, ale současně nejsou považovány za chybu vstupu. Grafy, které mají lichý počet vrcholů, není možné rozdělit do párů a proto takové zadání bude vždy neřešitelné. Server tuto podmínku záměrně nekontroluje, uživatel nesplnitelnost zjistí pomocí generované formule v řešiči SAT.

Původně bylo zamýšlena implementace pouze pro bipartitní grafy. Vstupní graf však není kontrolován, zda je bipartitní, a řešení funguje správně pro libovolný graf.

[1] HTML5 validace probíhá v enginu prohlížeče, jenž definuje texty vlastních chybových hlášek pro různé elementy stránek a ty vypíše před textem z atributu `title`. V prohlížečích používající češtinu může dojít ke smíchání s anglickým textem (např. „Vyplňte prosím pole v požadovaném formátu: Insert 1 to 3 alphanumeric characters.“). Chybové hlášení z prohlížeče se zobrazí také v případě, že atribut `title` chybí.

4.7.3 Součet podmnožiny

Webový formulář obsahuje na prvním řádku pole pro zadání součtu (Sum) a každý další řádek reprezentuje jedno číslo (64 bit unsigned integer) z množiny všech čísel, jež u problému uvažujeme. Také u tohoto problému řeší validaci HTML5. Správný formát čísla ověřuje regulární výraz:

```
[0-9]{1,19}
```

Číslo tedy musí mít maximálně 19 znaků.

Textový soubor sestává z libovolného počtu řádků, každý obsahuje samostatné číslo, přičemž první řádek je chápán jako požadovaný součet.

```
125
1
2
4
8
16
32
64
128
```

4.8 Generování formulí

Formule jsou generovány tak, aby byly co nejpodobnější teoretickým popisům zobrazovaným na serveru a také uvedených zde v kapitole 3. Uvedeme si příklad pro část formule z problému sudoku, jež je vyjádřena následujícím vzorcem:

$$\varphi_{rows} = \bigwedge_{i=1}^{9} \bigwedge_{k=1}^{9} \varphi_{row}^{i,k}$$

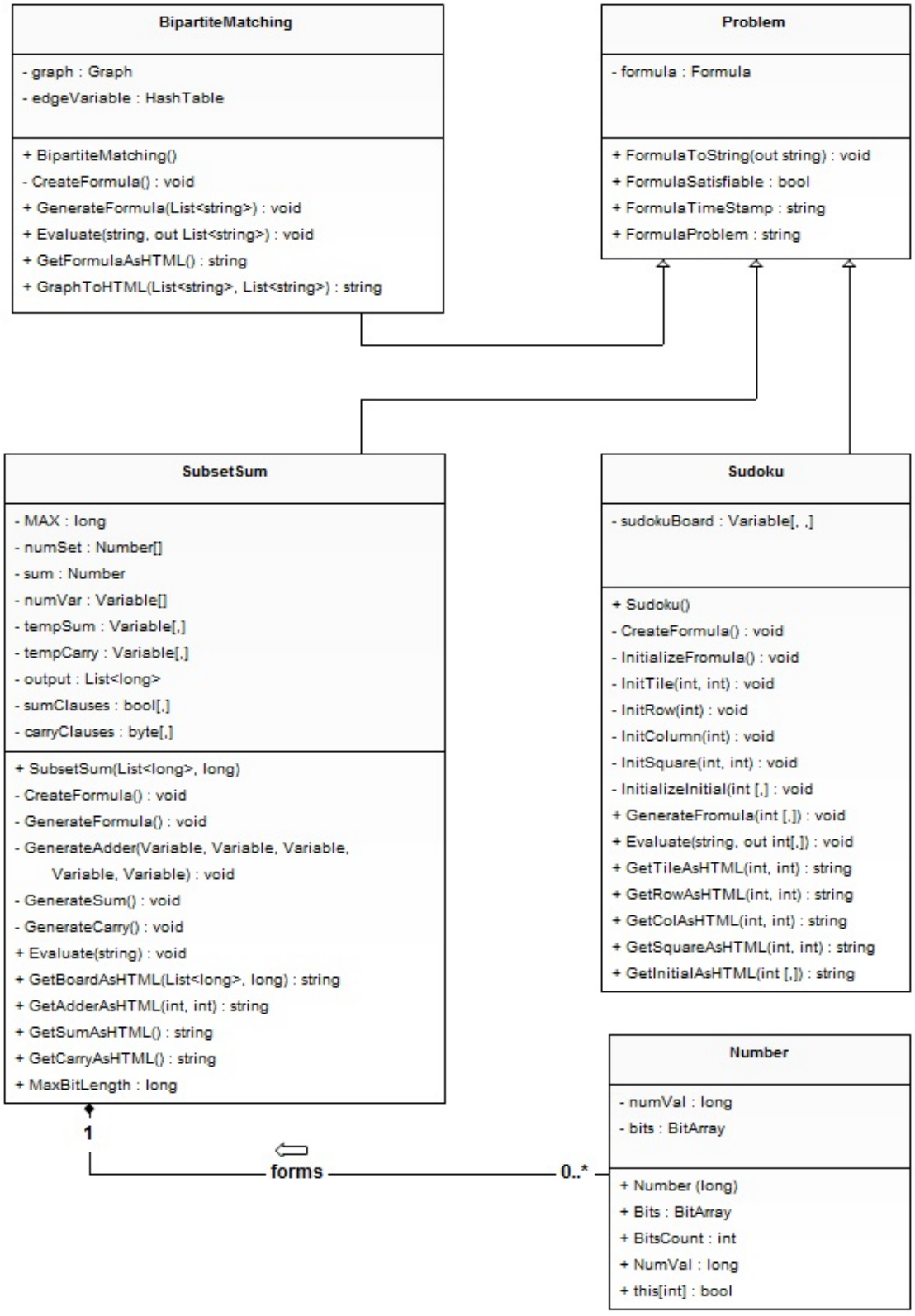
$$\varphi_{row}^{i,k} = (x_{i1k} \vee x_{i2k} \vee x_{i3k} \vee x_{i4k} \vee x_{i5k} \vee x_{i6k} \vee x_{i7k} \vee x_{i8k} \vee x_{i9k}) \wedge \bigwedge_{m=1}^{9} \bigwedge_{n=m+1}^{9} (\neg x_{imk} \vee \neg x_{ink})$$

Tuto část generuje následující kód:

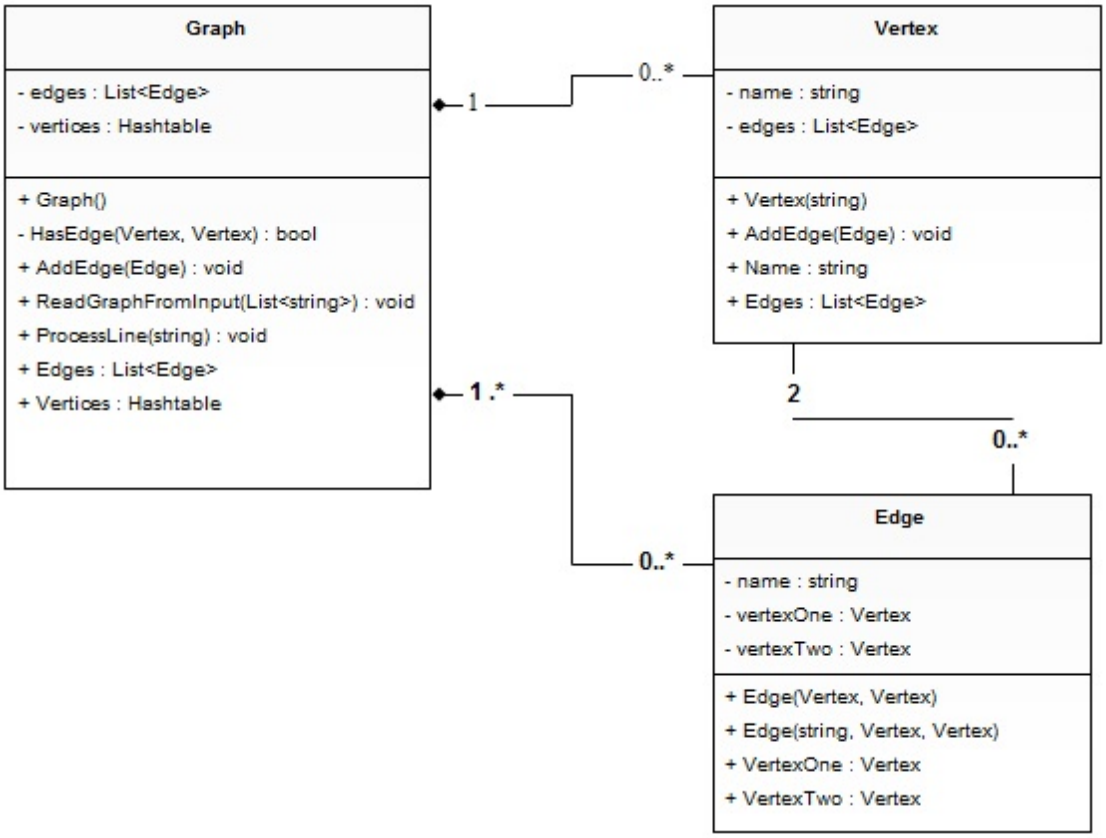
```
for (int i = 0; i < 9; i++)
{
    Clause c = new Clause();

    for (int j = 0; j < 9; j++)
    {
        c.AddLiteral(sudokuBoard[row, j, i ]. Positive );
    }

    formula.AddClause(c);

    for (int j = 0; j < 9; j++)
```

```
        {
            for (int k = j + 1; k < 9; k++)
            {
                Clause c2 = new Clause();

                c2.AddLiteral(sudokuBoard[row, j, i].Negative);
                c2.AddLiteral(sudokuBoard[row, k, i].Negative);

                formula.AddClause(c2);
            }
        }
    }
```

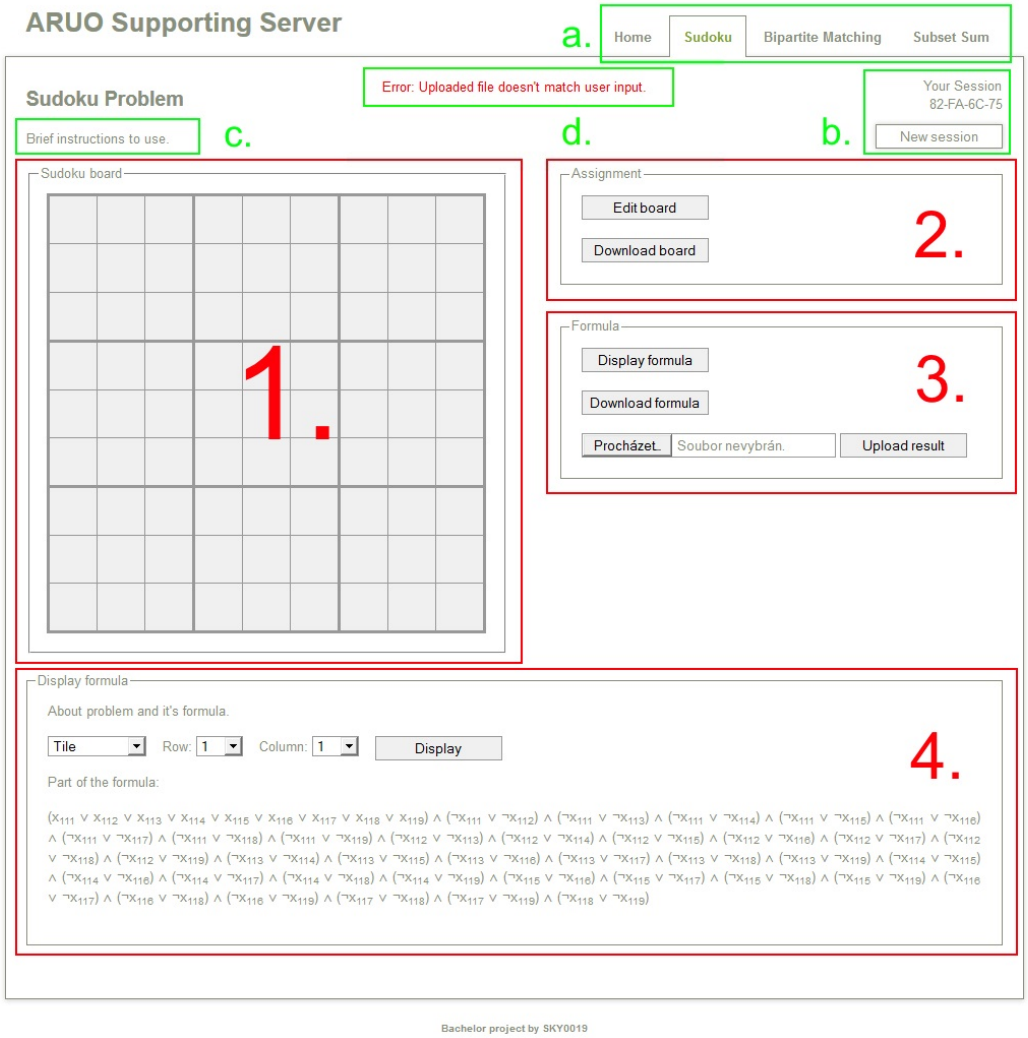
Tento kód je volán pro každý řádek. Hodnotu tohoto řádku dostaneme na vstupu v parametru `row`. Vnější cyklus iteruje přes čísla `i`. První vnitřní cyklus generuje jednu klauzuli, ve které se nachází disjunkce všech pozitivních literálů proměnných pro daný řádek `row`, dané číslo `i` v různých sloupcích `j`. Další dva do sebe zanořené cykly generují klauzule negativních literálů na daném řádku `row`, s daným číslem `i` a postupně všemi možnými dvojicemi sloupců `j` a `k`. Správně utvořená klauzule je vždy přidána do kolekce všech klauzulí pomocí metody `AddClause(Clause c)` objektu třídy `Formula`.



Obrázek 4: Třídní diagram 2: Typy problémů



Obrázek 5: Třídní diagram 3: Reprezentace grafu



Obrázek 6: Struktura uživatelského rozhraní

5 Další možnosti rozšíření

Při vývoji byla snaha řešit výukový server modulárně, je zde tedy prostor pro další možnosti rozšíření:

1. Řešení dalších samostatných problémů, tedy doplnění nových modulů.
2. Zpracování zobrazení grafu u problému úplného párování v graficky sofistikovanějším provedení (například uživateli poskytnout možnost kreslit graf přímo v prohlížeči).
3. Aplikace podporuje v současné době generování formule pro řešič SAT pouze ve formátu DIMACS CNF. Možným rozšířením by mohlo být doplnění dalších formátů pro generované formule.
4. V rámci jmenného prostoru `SolverFoundation.Solver` pro jazyk C# je k dispozici třída `SatSolver`, potenciálně by tedy bylo možné vyvinout build-in řešič SAT přímo do aplikace.
5. Implementace logiky pro kontrolu výskytu duplicitních klauzulí po redukci popsané v podkapitole součet podmnožiny 3.3.
6. Doplnění alternativních způsobů vkládání ohodnocení z řešiče SAT (například vložení textu do formuláře).

6 Závěr

V rámci zadání byl implementován server na podporu výuky předmětu ARUO. Server byl vyvinut na platformě ASP.NET v jazyce C# a v současnosti poskytuje rozhraní pro tři různé problémy. Pro každý problém může uživatel zadat vstup, zobrazit formuli nebo její část v matematické notaci přímo na stránce, stáhnout textový soubor s formulí v formátu DIMACS CNF a nahrát na server textový soubor s evaluací z řešiče SAT pro zobrazení řešení. Kromě toho je pro každý problém napsán teoretický popis částí formule, aby si jej uživatel snadno vyložil. Zadání každého problému lze uložit a načíst z textového souboru.

Jelikož práce splňuje zadání, bude server využíván při výuce a je plánováno jeho další rozšíření v rámci navazujících studentských prací.

7 Přílohy

Veškeré přílohy jsou uloženy na přiloženém CD. Jednotlivé složky obsahují:

- `application` - vlastní aplikace
  - `release` - ZIP archív se soubory potřebnými pro nasazení aplikace na serveru
  - `source` - ZIP archív s kompletním projektem z programu Visual Studio 2013
- `diploma` - text bakalářské práce
  - PDF verze
  - `source` - zdrojové soubory pro LaTeX včetně použitých obrázků a Makefile
- `examples` - textové soubory s testovacími příklady vstupů pro jednotlivé problémy, ke každému je k dispozici i generovaná formule a získané ohodnocení z řešiče SAT
- `programmer-manual` - programátorská příručka vygenerovaná programem Doxygen z komentářů ve zdrojovém kódu aplikace
  - PDF verze
  - `html` - HTML verze
- `user-manual` - uživatelská příručka
  - PDF verze
  - `source` - zdrojové soubory pro LaTeX včetně použitých obrázků

8 Reference

[1] *COOK, Stephen A. The complexity of theorem-proving procedures.* Proceedings of the third annual ACM symposium on Theory of computing - STOC '71, New York, USA: ACM Press, 1971, s. 151-158. DOI: 10.1145/800157.805047.

[2] *The international SAT Competitions* [online]. 2014 [cit. 2015-05-02]. Dostupné z: `http://www.satcompetition.org/`.

[3] *The MiniSat Page* [online]. 2005 [cit. 2015-05-02]. Dostupné z: `http://minisat.se/MiniSat.html`.

[4] *Cygwin* [online]. 2013 [cit. 2015-05-03]. Dostupné z: `https://www.cygwin.com/`.

[5] *MiniSat in your browser* [online]. 2013 [cit. 2015-05-02]. Dostupné z: `http://www.msoos.org/2013/09/minisat-in-your-browser/`.

[6] *DUŽÍ, Marie. Logika pro informatiky (a příbuzné obory): učební text* [online]. Ostrava: VŠB-TU Ostrava, 2012, č. 1, s. 179 [cit. 2015-05-02]. Dostupné z: `http://www.cs.vsb.cz/duzi/Matlogika_ESF_Definite.pdf`.

[7] *MiniSAT User Guide: How to use the MiniSAT SAT Solver* [online]. 2008 [cit. 2015-05-02]. Dostupné z: `http://www.dwheeler.com/essays/minisat-user-guide.html`.

[8] *KOVÁŘ, Petr. Teorie grafů: učební text* [online]. Ostrava: VŠB-TU Ostrava, 2013, č. 1, s. 216 [cit. 2015-05-03]. Dostupné z: `http://homel.vsb.cz/~kov16/files/skriptum_teorie_grafu_rozsirene_tisk.pdf`.

[9] *Třídy složitosti a Turingovy stroje - Algoritmy.ne* [online]. 2012 [cit. 2015-05-02]. Dostupné z: `http://www.algoritmy.net/article/5774/Tridy-slozitosti`.

[10] *ASP.NET Open Source: ASP.NET is part of a great open source .NET community* [online]. 2015 [cit. 2015-04-13]. Dostupné z: `http://www.asp.net/open-source`.

[11] *ASP.NET Web Forms: lets you build dynamic websites using event-driven model and hundreds of controls and components* [online]. 2015 [cit. 2015-04-26]. Dostupné z: `http://www.asp.net/web-forms`.

[12] *Martin Fowler: MVP, Passive View* [online]. 2006 [cit. 2015-04-28]. Dostupné z: `http://martinfowler.com/eaaDev/PassiveScreen.html`.

[13] *Source Making: Observer Design Pattern* [online]. 2015 [cit. 2015-04-28]. Dostupné z: `https://sourcemaking.com/design_patterns/observer`.

[14] *Colorbox - a jQuery lightbox* [online]. 2015 [cit. 2015-05-01]. Dostupné z: `http://www.jacklmoore.com/colorbox/`.

[15] *GasTeX: Graphs and Automata Simplified in TeX* [online]. 2014 [cit. 2015-05-01]. Dostupné z: http://www.lsv.ens-cachan.fr/~gastin/gastex/.

[16] *LaTeX2HTML Development Repository* [online]. 2001 [cit. 2015-05-01]. Dostupné z: http://www.latex2html.org/.

[17] *SatSolver Class (Microsoft.SolverFoundation.Solvers)* [online]. 2015 [cit. 2015-04-28]. Dostupné z: https://msdn.microsoft.com/en-us/library/microsoft.solverfoundation.solvers.satsolver%28v=vs.93%29.aspx.